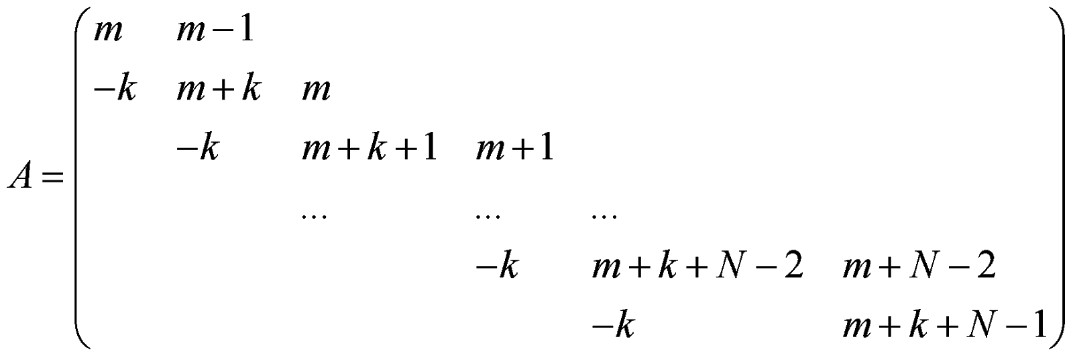
**Лабораторная работа 3**

**«Решение СЛАУ методом прогонки»**

Необходимый для выполнения работы теоретический материал и формулы имеются в файле «Метод прогонки».

**Задание.** Разработать программу численного решения методом прогонки СЛАУ вида *Ay=f* (вида (1) в файле «Метод прогонки»).

Матрицу системы задать следующим образом:

*.*

Правую часть *f* задать умножением матрицы *A* на вектор *y=*(1, 2, ... , *N*+1):  *f=Ay*.

Для вычислений выбрать параметры:

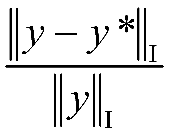
* *m* – номер в списке студенческой группы;
* *N*+1  – (порядок матрицы) одно из чисел в пределах от 10 до 12;
* *k* – номер студенческой группы.

Программно реализовать (в качестве языка программирования выбрать C или C++) вычисления для рассматриваемого примера. Для вычислений использовать тип float.

В отчете должны быть представлены:

1. Преобразованная матрица после прямой прогонки

2. Вектор приближённого решения *y\**.

3. Относительная погрешность вида , где *y* – точное решение.

Содержание работы должно включать следующие пункты.

1. Постановка задачи.
2. Входные данные.
3. Листинг программы. Обязательны подробные комментарии.
4. Выходные данные.
5. Выводы.

**Входные данные**

n=12, m=13, k=3

**Листинг программы**

#include <iostream>

#include <random>

#include <cmath>

#include <iomanip>

using namespace std;

//Размерность матрицы

int n = 12;

//Номер в списке группы

int m = 13;

//Номер группы

int k = 3;

//Нахождение случайного числа

int Rand(int L, int R) {

static random\_device rd;

static mt19937 gen(rd());

uniform\_int\_distribution<> dis(L, R);

return dis(gen);

}

//Вывод матрицы

void PrintMatrix(float\*\* A)

{

cout << "A:" << endl;

for (int i = 0; i < n; ++i)

{

for (int j = 0; j < n; ++j)

{

cout << fixed << setprecision(3) << setw(10) << A[i][j] << " ";

}

cout << endl;

}

cout << endl;

}

//Вывод столбца

void PrintArray(float\* x)

{

for (int i = 0; i < n; ++i)

cout << fixed << setprecision(7) << setw(10) << x[i] << " ";

cout << endl << endl;

}

//Умножение матрицы на столбец

void Multiplication(float\*\* A, float\* x, float\* b)

{

float s;

for (int i = 0; i < n; ++i)

{

s = 0;

for (int j = 0; j < n; ++j)

{

s += A[i][j] \* x[j];

}

b[i] = s;

}

}

void Progonka(float\*\* A, float\* f)

{

A[0][1] /= A[0][0];

f[0] /= A[0][0];

A[0][0] = 1;

for (int i = 1; i < n - 1; ++i)

{

A[i][i + 1] /= A[i][i] - A[i - 1][i] \* A[i][i - 1];

f[i] = (f[i] - A[i][i - 1] \* f[i - 1]) / (A[i][i] - A[i - 1][i] \* A[i][i - 1]);

A[i][i] = 1;

A[i][i - 1] = 0;

}

f[n - 1] = (f[n - 1] - A[n - 1][n - 2] \* f[n - 2]) / (A[n - 1][n - 1] - A[n - 1][n - 2] \* A[n - 2][n - 1]);

A[n - 1][n - 1] = 1;

for (int i = n - 2; i >= 0; --i)

{

f[i] -= f[i + 1] \* A[i][i + 1];

}

}

//Нахождение относительной погрешности

float Fault(float\* x, float\* b)

{

float\* dx = new float[n];

for (int i = 0; i < n; ++i)

dx[i] = x[i] - b[i];

float ndx = 0;

for (int i = 0; i < n; ++i)

ndx += dx[i] \* dx[i];

ndx = sqrt(ndx);

float nx = 0;

for (int i = 0; i < n; ++i)

nx += x[i] \* x[i];

nx = sqrt(nx);

cout << fixed << setprecision(10) << "Относительная погрешность: " << 100 \* ndx / nx << "%" << endl << endl;

return ndx / nx;

}

int main()

{

setlocale(LC\_ALL, "rus");

//Создание матрицы n\*n

float\*\* A = new float\* [n];

for (int i = 0; i < n; ++i)

{

A[i] = new float[n];

}

//Инициализация

for (int i = 0; i < n; ++i)

for (int j = 0; j < n; ++j)

A[i][j] = 0;

A[0][0] = m;

A[0][1] = m - 1;

for (int i = 1; i < n; ++i)

for (int j = 0; j < n; ++j)

{

if (j == i - 1)

A[i][j] = -k;

else if (j == i)

A[i][j] = m + k + i - 1;

else if(j == i + 1)

A[i][j] = m + i - 1;

}

PrintMatrix(A);

float\* y = new float[n];

for (int i = 0; i < n; ++i)

y[i] = i + 1;

PrintArray(y);

float\* f = new float[n];

Multiplication(A, y, f);

PrintArray(f);

Progonka(A, f);

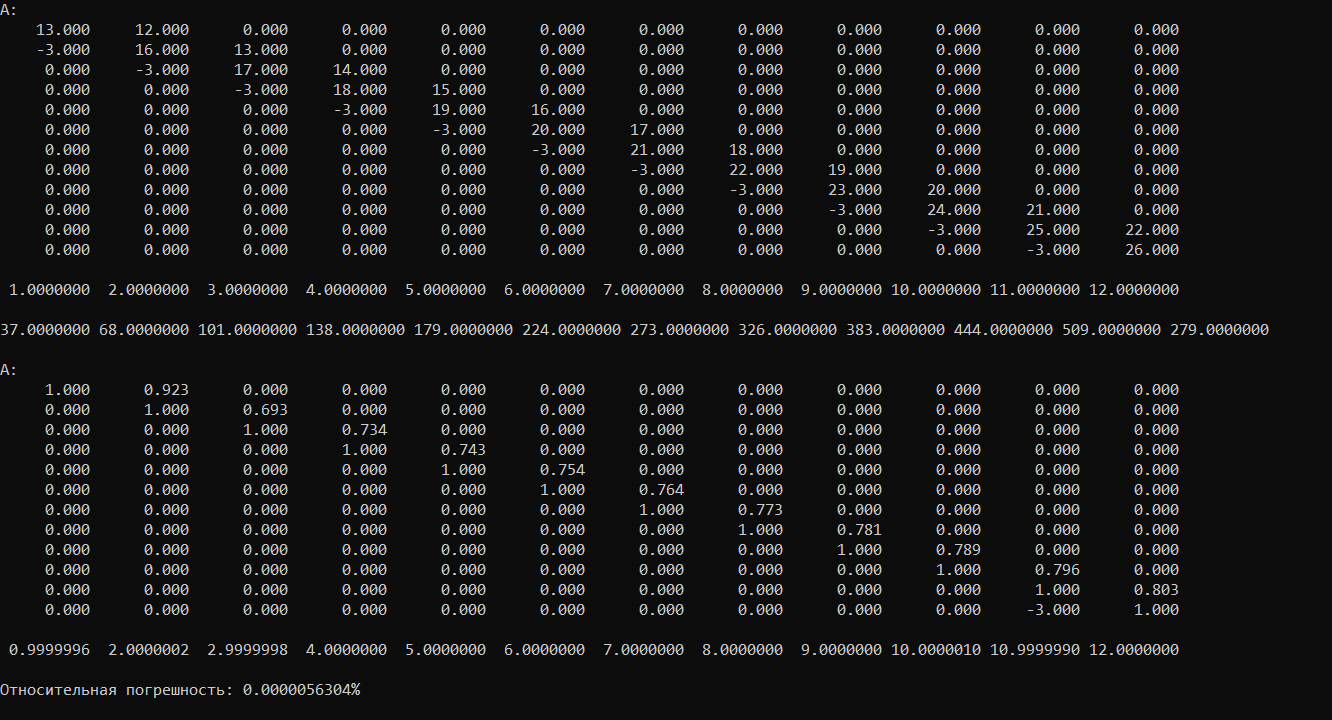
PrintMatrix(A);

PrintArray(f);

Fault(y, f);

}

**Выходные данные**

****

**Вывод**

Метод прогонки является прямым методом, его точность сопоставима с точностью метода Гаусса, однако метод прогонки можно применить только на трехдиагональных матрицах или на матрицах с диагональным преобладанием.